Effect of Noise on Generalisation in Massively Parallel Fuzzy Systems

Sameer Singh

School of Computing, University of Plymouth, Plymouth PL4 8AA, United Kingdom

Singh, S. "Effect of noise on generalisation in Massively Parallel Fuzzy Systems", Pattern Recognition, vol. 31, issue 11, pp. 25-33, 1998 .

Effect of Noise on Generalisation in Massively Parallel Fuzzy Systems

Abstract

This paper studies the performance of Massively Parallel Fuzzy Systems (MPFS) on the two spiral benchmark. Spiral data is contaminated with five different noise distributions. The recognition rates of the system are reported with varying levels of different types of noise. The behaviour of the system is investigated with additive, multiplicative, cumulative and non-cumulative noise. The results show that the MPFS system remains stable to different noise variations and the generalisation error remains consistently low. As the total noise in the system increases, the system witnesses a linear decrease in entropy and the generalisation error is easier to predict. The error rate is found to have two separate patterns of variation for cumulative and non-cumulative noise.

Keywords: MPFS, noise distribution, possibility, recognition rate, spiral benchmark

1 INTRODUCTION

Massively Parallel Fuzzy Systems (MPFS) were first proposed by Singh to solve pattern recognition problems in real-time. The initial work with recognising manufacturing assembly data showed that the proposed system was a powerful classifier capable of recognising patterns in real-time and it was suggested that such mechanisms may be adopted in knowledge-based systems for searching rules [1] or as independent modules which determine the nearest neighbour in training data [2]. The main advantage offered by the system as a decision making tool is their speed of operation and computational cheapness. Evidently, these systems are capable of performing pattern recognition and decision making for any problem domain, at least in theory. Since their early publication, MPFS has been tested on two more applications: electronic nose odour recognition and the two spiral benchmark. Singh et al. [3] find that an MPFS architecture supported by possibility generation rules [4, 5] is a powerful classifier for classifying coffee data and gives an intermediate performance between non-fuzzy backpropagation neural networks and fuzzy neural networks [6]. This is the only study on the performance superiority of MPFS systems over non-fuzzy neural networks. MPFS has also been extensively tested on the two spiral benchmark* and very encouraging results have been obtained. It has been observed that an MPFS can be used to recognise the benchmark with 100% recognition rate on validation data and 99.6% on test data (submitted work).

The MPFS architecture consists of three primary components: a set of processors P which are responsible for calculating the possibility that a particular test pattern belongs to class k (k = 1... N), a decision making module M which takes the output from individual processors and assigns the test pattern to one of the k classes, and a set of counters C that monitor the total number of correctly predicted patterns (Figure 1). The input data is presented to the processors sequentially and at any time t, each processor receives the same test pattern p_Γ for analysis. The processor P_k computes the possibility $\rho_k(p_\Gamma)$ that the test pattern $p_\Gamma = (x_{k1,}\ x_{k2\ \ldots}\ x_{kn})$ belongs to class k. For this, individual datum possibilities in a pattern are combined together with a ξ function (min-max or product operator). These operators have been traditionally used in the fuzzy theory [5]. The MPFS system is restricted to using the same possibility calculation method across all processors during input presentation. The processor P_k with the highest possibility claims the test pattern, i.e. pattern p_Γ belongs to class k. The set of counters C monitor the total number of correct classifications and thus indicate the error rate.

***** Two spiral benchmark is available at the AI repository at Carnegie Mellon University

Figure 1

For parametric data, possibilities may be calculated using a quadratic function as detailed by Zadeh[5] and Mamdani & Gaines[4]. Such functions approximate a Gaussian function for a large amount of training data. For non-parametric and temporal data, data density functions are not well defined including validation methods [7]. In such cases, novel functions and validation methods must be rigorously sought. This is especially important for temporal data which may be classified into two types: temporal data for which data is a function of time and the higher order function itself is fixed; temporal data generated by processes which themselves evolve with time. The latter type has not been studied in detail and yet demands the most attention.

MPFS is a non-iterative system. The training data for class *k* is used to compute the possibility that test pattern p_Γ belongs to it. Hence, different processors work with different partitions of the training data, i.e. processor P_k only works with the training data partition for class *k* (C_k). This has an important advantage of reducing computational load on processors. In addition, for a total of N classes the computational time reduces to 1/N times the total time taken for serial processing. The MPFS system can be used with traditional validation procedures such as cross-validation and bootstrapping [8]. However, for temporal data, removal of training data for test purposes can decimate the temporal information encoded in data and produce false estimates. For this, further research is needed to develop rigorous validation procedures for temporal data to give an estimate of the true error [7]. In Fig. 1, the dotted lines represent any feedback connection if applicable for the problem. MPFS systems can therefore incorporate dynamic learning by including correct predictions in the training data at the testing proceeds, i.e. test decisions are based on a dynamically enlarging training set from which the system can be trained regularly.

The main aim of this paper is to study the generalisation performance of a Massively Parallel Fuzzy System with noisy non-linear data. Such a study is important for two reasons: to document the stability and reliability of the system with increasing noise of different underlying distributions, and to illustrate system behaviour. In this paper, the performance of the system is tested on the two spiral benchmark. This particular benchmark was chosen because: a) spiral data is highly non-linear and temporal in nature; b) a considerable amount of previous work exists on the subject to make valid comparisons; c) spiral data exists in several scientific applications and its recognition is of practical importance [9]; and d) neural networks using backpropagation and its relatives encounter significant convergence problems [10, 11, 12] and the trained system is underconstrained [13]. It has been realised in the past that recognising the

benchmark itself is not sufficient for a classifier: in addition, it should work in real-time and be resistant to random noise variations. The latter aspect has not been studied in detail as yet on a noisy spiral benchmark. However for real-time spiral recognition, previous work includes the development of minimal configuration neural network [14], data encoding methods [15, 16], a neurofuzzy classifier [17], a fast learning ANN [18], and hypercube separation algorithm [19]. This paper studies the real-time performance of MPFS on spiral data and tests their stability and behaviour with injected noise with different underlying distributions. The inner working of processors responsible for possibility calculation is first explained.

2 FUZZY SPIRAL PATTERN RECOGNITION

The two spiral benchmark consists of data for two spirals that coil around each other and around the origin. The spirals are generated with three primary variables: density, radius and offset. These three variables for the original benchmark default to 1, 6.5 and 0.1 respectively but may be changed to suit specific needs.

Figure 2

The radius σ of the spiral specifies the maximum radius since the actual radius changes with time. The density of the spiral determines the total number of points generated, i.e. its length. The benchmark used in its original form here consists of a total of 194 points in the input space, 97 in each of the two classes. The task of a classifier is to differentiate between these two spirals whose {**X**, **Y**} coordinates are given. The classifier should be able to recognise the true class of patterns in the validation and test sets which are generated by offsetting training data by a factor δ in opposite directions.

It is important to remember that spiral data is temporal. Hence, Gaussian techniques used conventionally for estimating possibilities [5] are not well suited: this was evident from initial experimental results using MPFS. Hence, it was decided to determine the nearest neighbour of the test pattern in training data. The overall procedure is explained below:

The data is initially divided as of training and test type. The purpose of the classifier is to allocate a class to the test data on the basis of training data on which it has been trained. The performance of the classifier is measured on the basis of correct test predictions that it makes. For a given test pattern X, the fuzzy

classifier computes the membership of X in different classes C_1, ... C_j ... C_m where 1 • j • m. The membership of X in class C_j can be expressed as $\mu_j(X)$. The test pattern is allocated to a class for which the membership function yields the maximum value. The overall process may be mathematically explained as: Consider an unknown pattern X represented by a point in a multi-dimensional space $\bullet_X$ consisting of *m* pattern classes C_1 ... C_m. Let R_1 ... R_j ...R_m be the reference vectors where R_j associated with C_j contains h_j number of prototypes such that, $R_j^{(l)} \in R_j$, l = 1, 2, ... h_j

$$\mu_j^l(X) = [1 + \{d(X, R_j^{(l)})/F_d\}^{F_e}]^{-1.0} \quad \text{... (1)}$$

where $\mu_j^l(X)$ is the membership of X in class C_j as determined through the class sample *l*, and $d(X, R_j^l)$ is the distance between X and R_j^l. In eqn. (1), F_e and F_d are positive constants that determine the degree of fuzziness in the membership space. The main purpose of using the membership function is to map an *n*-dimensional feature space into a m-dimensional membership space which is a unit hypercube and satisfies the following conditions:

$$\mu_j^l(X) \rightarrow 1 \text{ as } d(X, R_j^l) \rightarrow 0 \quad \text{... (2)}$$

$$\mu_j^l(X) \rightarrow 0 \text{ as } d(X, R_j^l) \rightarrow \bullet \quad \text{... (3)}$$

$$\mu_j^l(X) \rightarrow \text{increases, as } d(X, R_j^l) \rightarrow \text{decreases} \quad \text{... (4)}$$

The test pattern X is allocated to class *i* if,

$$\mu_i(X) \geq \mu_j(X) \text{ for } i \neq j, \text{ where } i, j = 1... m, \text{ and } \mu_j(X) = \max \sum_{l=1}^{h_j} (\mu_j^l(X)) \quad \text{... (5)}$$

In this manner, the class of X is the class for which it has the highest membership value.

This approach is in a way based on finding the nearest neighbour of the test pattern in the training data. The class of the nearest neighbour is assigned as the class of the test pattern. The method of this class assignment is however considerably different from the well known “K nearest neighbour method”. The established k-NN method: “... involves finding a hypersphere around a point X which contains K points (independent of their class), and then assigning X to the class having the largest number of representatives inside the hypersphere”, Bishop [7, p. 57]. This method however achieves limited success with the spiral data as shown in Figure 2. Here, generating a spherical boundary around a test pattern (black square) encloses equal number of class C_1 patterns (white circles) and C_2 patterns (black circles).

Previous results with this approach show between 50 to 55% correct classification of spiral data. In the proposed method however, only one nearest neighbour is found from the training set. The quality of the classification would therefore depend on the sensitivity of the method for detecting the true nearest neighbour. The fuzzy approach to classification is generic to any test domain, but possibility calculation methods may need modification in different domains, e.g. manipulating the fuzzy parameters F_d and F_e in eqn. 1. One of the important advantages of the fuzzy pattern recognition method is that possibilistic algorithms can perform decision making with equal or weighted role of input features and with rejection thresholds where decisions based on low possibility counts are discarded, Bishop [7, p. 28].

Similar to the k-NN method, the Gaussian approach has serious limitations when solving the 2D spiral problem. The main difficulty lies with the procedure used for calculating class memberships. The spiral data symmetrically coils around the origin and for either classes, *x* and *y* measurement means are 0. Possibility calculations using the traditional approach would be therefore less sensitive to the temporal nature of the helix and a number of test patterns may have equal memberships in different classes. In order to solve a spiral problem, the procedure for membership computation is explained below:

❶ Label the data as for training set Ω, validation set V and test set T. The benchmark consists of N =194 patterns in each of these sets.

❷ Separate class C_1 data and class C_2 training data in two training files F1 and F2.

❸ For every pattern in $p_V = (x_v, y_v)$ in the validation set V, perform the following steps:

❹ Find the upper and lower bounds of x_v for class C_1 from F1. These may be represented as $x_v(lb_1)$ and $x_v(ub_1)$. Here lb_1 and ub_1 are positions at which lower and upper bound are found in the training data array Ω for class C_1. If $x_v > x_\Omega$ for all $x_\Omega \in$ F1, then x_v has only a lower bound. Similarly, if $x_v < x_\Omega$ for all $x_\Omega \in$ F1, then x_v has only an upper bound.

If C_1 has a total of h samples, the upper bound of x_v in C_1 is x_i such that

$x_i - x_v < x_k - x_v$ for all $x_i \geq x_v$ and $x_k \geq x_v$, $x_i \in C_1$, $x_k \in C_1$, $1 \leq i \leq h$, $1 \leq k \leq h$, $i \neq k$. Similarly, a lower bound can be found.

❺ For F1, calculate class membership for the following cases:

Case1

p_v already exists in F1 as a class C_1 pattern : $\mu_1(x_v) = 1.0$; : $\mu_1(y_v) = 1.0$;

Case2

x_v exists in F1 at position i, $1 \leq i \leq N$ but not y_v in the same position.

$\mu_1(x_v) = 1.0$; $\mu_1(y_v) = 1.0/(1.0 + \eta_1)$ where $\eta_1 = |y_v - y_j|$

$\text{Lt}\ \eta_1 \to 0\ \ \mu_1(y_v) = 1.0$;

Case3

y_v exists in F1 at position j but not x_v in the same combination

$\mu_1(y_v) = 1.0$; $\mu_1(x_v) = 1.0/(1.0 + \eta_2)$ where $\eta_2 = |x_v - x_j|$

$\text{Lt}\ \eta_2 \to 0\ \ \ \mu_1(x_v) = 1.0$;

Case4

x_v and y_v do not occur in F1

$\mu_1(x_v) = 1.0/(1.0 + \eta_3)$ if $\eta_3 \leq \eta_4$ else

$\mu_1(x_v) = 1.0/(1.0 + \eta_4)$ if $\eta_4 < \eta_3$

where $\eta_3 = |x_v - x_v(lb_1)|$ and $\eta_4 = |x_v(ub_1) - x_v)|$

$\mu_1(y_v) = 1.0/(1.0 + \eta_5)$ if $\eta_5 \leq \eta_6$ else

$\mu_1(y_v) = 1.0/(1.0 + \eta_6)$ if $\eta_6 < \eta_5$

where $\eta_5 = |y_v - y_v(lb_1)|$ and $\eta_6 = |y_v(ub_1) - y_v)|$

❻ Perform steps 4 & 5 on F2 to calculate $\mu_2(x_v)$ and $\mu_2(y_v)$.

❼ Derive an optimal function ξ for the following:

If $\xi(\mu_1(x_v), \mu_1(y_v)) > \xi(\mu_2(x_v), \mu_2(y_v))$ then $p_v \in C_1$, else $p_v \in C_2$.

The results shown later have used the multiplication function for possibility combination.

❽ Determine the recognition rate through correctly classified patterns.

❾ Test the approach on the test set T by following steps 2 to 8.

In the above discussion, the membership μ in case 2 is inversely proportional to the distance between the target and the actual *y* value in the same position. In case 4 when both *x* and *y* test values are not present in the training set, the membership is inversely proportional to the distance between the test value and either its upper or lower bounds depending on the nearest one in the training file. It may be observed that in the described method, $F_e = 1$ and $F_d = 1$. In the above algorithm, if we find x_v or y_v in F1

or F2 in more than one position, the membership function at these different positions is calculated and the highest value is chosen. In rare cases it is also possible that x_v or y_v may have two or more upper and lower bounds at different positions in F1 or F2. As previously, the membership function is computed for these different cases and the highest value is chosen to be $\mu_1(x_v)$ or $\mu_1(y_v)$ as may be the case.

3 NOISE INJECTION IN SPIRAL DATA

In the past, noise injection has been studied from three perspectives: its effect on a) generalisation and recognition rates of classifiers; b) convergence of iterative classifiers; and c) stability of the classifier. From the point of view of our discussion, only a) and c) are relevant. Noise effects have been mostly studied for neural networks in the past than for fuzzy classifiers, for example see [20, 21, 22]. Although neural networks as pattern classifiers have an altogether different approach than MPFS, the above studies are considerably important to support some of the conclusions of this study. In addition, this work draws upon the approaches used in these previous studies, especially [21], to follow traditional guidelines.

For an MPFS architecture, noise injection can have two main effects: a) high levels of noise may destabilise the system and make it unpredictable; and/or b) it may lead to high error rates and poor generalisation. System stability and generalisation performance will depend on: the quality of noise which is primarily determined by its underlying distribution, and its quantity which is determined by the maximum allowed offset, the type of noise - additive or multiplicative, cumulative or non-cumulative, and the noise generation parameters including its variance. This paper studies the effect of noise on the spiral test data recognition performance. The spiral training data is free of noise. The validation and test data is created by injecting training data with high levels of noise. In the original spiral benchmark, validation and test data are both generated by displacing training data using an offset of $\delta = 0.1$ in opposite directions. For this study, each pattern is displaced by a different amount depending upon the level of pseudo-random noise. The maximum amount by which a test or validation pattern may be displaced by additive noise is given by $\mathbf{r}*\delta$ where r is a random noise value within the [0, 1] range and δ may be varied. Thus, if the training data is represented as $\{\mathbf{X}, \mathbf{Y}\}$, the validation set is $\{\mathbf{X}, \mathbf{Y} + \mathbf{r}*\delta)$ and the test set is $\{\mathbf{X} + \mathbf{r}*\delta, \mathbf{Y})$ for additive noise. The injected noise could also be multiplicative: the validation set $\{X, Y(1 + \mathbf{r}*\delta)\}$, and the test set $\{\mathbf{X}(1 +\mathbf{r}*\delta), \mathbf{Y}\}$. In addition, the noise may be non-cumulative where previous noise variations have no effect on the current test data and in turn on the decision making, or cumulative where the noise at time t is a cumulative sum of previous noise variations starting at $t = 0$.

The quality of noise primarily determines the vector **r**. Here, noise is generated using C++ library functions for five continuous noise distributions whose characteristics are summarised in Table 1 and their description may be found in Canavos[23].

Table 1

In Table 1, all random noise output is linearly translated to the [0, 1] range except Gaussian noise which may be negative and is translated within the [-1, +1] range. All distributions are generated with variance $\sigma^2 = 1.0$ and variable mean. The multiplication of the maximum offset δ and vector **r** produces higher amplitude of added noise for higher values of δ.

4 PERFORMANCE ANALYSIS

The performance of MPFS on spiral data may be discussed under two separate umbrellas: system behaviour, and generalisation ability. The first relates to how well the system recognises patterns and the rate of possibility change with time. The generalisation ability will be studied here by varying the quantity and quality of noise and observing the generalisation error. These are now discussed in turn.

4.1 System behaviour

- *Recognition Performance*: The MPFS spiral pattern recognition performance on spiral data with additive and multiplicative non-cumulative noise is shown in Table 2 and 3. First, Table 2 shows the recognition rate $\Re$ % with varying maximum offset δ and different noise distributions. The recognition rate % is the proportion of the total number of correctly predicted patterns to the total tested [7, 24]. For each different δ, noise is generated with a different seed so that a different set of random numbers are generated.

  Table 2

  The performance of the system is static in Table 2, i.e. all validation V and test T patterns are recognised using the training data Ω available at the beginning of the trial. There are several interesting performance

characteristics which may be noted from Table 2 with careful observation. The most obvious is that the MPFS system is considerably stable to additive noise variations of all types even with increasing levels as evident through high recognition rates. The poorest performance is observed with Gaussian noise, especially since negative noise is present and the best with negative exponential. In all types of distributions, higher levels of noise degrade system performance however in some cases there are random fluctuations when noise actually helps recognition. This last phenomenon has been observed in neural networks in previous studies that noise can actually help the train-test performance [21] and will be again observed later. Table 3 shows the system performance with multiplicative noise.

Table 3

As with additive noise, the system performance is stable and the recognition rates are high. Although recognition rates may on some occasions improve unpredictably, the general trend is a decline in performance with increasing δ. Fluctuations in performance are more pronounced, especially for Gaussian noise since noise is multiplicative. In general, for both additive and multiplicative noise, worst recognition rates are obtained with uniform and Gaussian noise. These two cases are now studied in more detail taking into account two system parameters: entropy ε and average total noise $\tilde{N}$.

- *Entropy and Noise*: "The entropy ε of a possibility based classifier can be defined as the total sum of possibilities generated by the system on a given data $\int_1^K \int_1^M (\rho)$ for a total of *K* classes, each consisting of *M* test patterns". For discrete data this may be represented as $\sum_1^K \sum_1^M (\rho)$. The entropy represents system decisiveness: higher ε represents the ability to distinguish between different classes of data and make good quality decisions (low possibilities generated by a classifier represent its inability to distinguish between overlapping classes and make good decisions [7]). This is very different from its traditional definitions where low entropy values represent system stability. For the spiral problem, ε_{max} = 582. The average total noise $\tilde{N}$ is the average total noise per pattern, i.e. $\tilde{N} = (\sum_{k=1}^{K} \sum_{m=1}^{M} \tilde{n}_{km})/ N$ where $\tilde{n}_{km}$ is the noise added to the m*th* pattern of class k, and N is the total number of patterns (N = K. M). Here $\tilde{N}$ only represents the mean noise per pattern although different test patterns are affected

by varying amounts depending on the maximum offset allowed and the sequence of random numbers generated. We observe that the variation in ε and $\tilde{N}$ for uniform noise as δ is varied. This is shown in Table 4.

Table 4

In Table 4, and 5, the behaviour of the system is shown for both additive and multiplicative non-cumulative noise for uniform and Gaussian noise respectively. Since *additive* noise is not multiplied by either of the **X**, **Y** vectors, the amount of noise added to the validation and test set is the same which is represented by $\tilde{N}(a)$. On the other hand, *multiplicative* noise varies for the validation and test set and is represented as $\tilde{N}(m)$. The following conclusions may be drawn from Table 4: a) the entropy ε of the system decreases almost linearly with increasing δ for both additive and multiplicative uniform noise; b) average $\tilde{N}(m)$ is more than twice $\tilde{N}(a)$ at almost each step and also increases linearly and; c) the entropy of the system and the noise present are inversely related. Table 5 shows the same phenomenon for Gaussian noise.

Table 5

In Table 5, the variation in noise change is more rapid and the drop in entropy more severe. Previous conclusions apply here too. In a few cases, the entropy drop is rapid whereas at other times there is a negligible or even an upward change when noise actually helps the recognition performance.

4.2 Generalisation ability

Till now the generalisation ability of the MPFS architecture has been evident only through its high recognition rates on the spiral data. This may be further investigated by: studying system performance on cumulative noise; and varying noise variance. This paper follows the methodology published by Jim et al. [21] for Gaussian and lognormal noise. As in their study, the effect of changing noise standard deviation on generalisation error is studied with the maximum permissible offset δ = 4.1. Figure 3 shows the plot for non-cumulative *multiplicative* NC(m) and non-cumulative *additive* NC(a) Gaussian noise affecting generalisation error. Figure 4 shows the same for lognormal noise.

The above plots demonstrate two important characteristics of MPFS systems working with the algorithm described in section 2: a) the variation in generalisation error with increasing noise standard deviation is alternating in nature and; b) generalisation error for multiplicative noise mirrors the one for additive noise but at a higher level. The alternating nature of the generalisation error can be explained due to the very nature of the fuzzy algorithm being used. With increasing standard deviation, the patterns most likely to be misclassified are offset sometimes closest to their correct class and often to the wrong class with the very next increase. The above plot has been generated on the same std. deviation range [0, 3] range as described by Jim et al. [21], However, these authors find a different relationship between the two variables studied for neural networks: for training noise in neural networks, the generalisation error first gradually decreases with increasing noise std. deviation and then rapidly increases after a minima. This process is certainly not evident in MPFS as seen above. We may now consider the relationship for cumulative noise (Figure 5 and 6).

Figure 5 and 6 demonstrate an important difference for cumulative noise compared to Figures 3 and 4: additive cumulative noise leads to higher error rates than multiplicative cumulative noise. This is directly opposite to the previous two plots where higher error rates are observed for multiplicative noise. Also, the error rates with cumulative noise are much higher as expected. There also seems to be a phase difference between the levels of error between additive and multiplicative noise with the previous lagging. The following conclusions may be summarised for both Gaussian and lognormal noise.

- additive noise is better tolerated by the system compared to multiplicative noise when it is non-cumulative, however, the opposite is true for cumulative noise.
- generalisation error shows alternating nature as standard deviation increases.
- generalisation error for multiplicative non-cumulative noise follows a similar trend to additive noise but at a higher level. The reverse is true for cumulative noise where there is a phase lag between the generalisation error with additive and multiplicative noise.
- generalisation error follows a flatter trend for non-cumulative noise than for cumulative noise.

In general, MPFS performance is considerably stable with the important advantage of real-time operation. The classification of a single pattern on Pentium 200 Mhz machine takes less than half a second including train-test time. This speed of recognition is important in several applications where MPFS systems should prove useful. Some examples where this system may be found attractive include real-time speech recognition, real-time image analysis, on-line character recognition, intelligent control and safety-critical applications, real-time sensor data analysis and robotics.

5 CONCLUSIONS

This paper has analysed the performance of Massively Parallel Fuzzy Systems on noisy spiral data. System behaviour and generalisation properties have been studied. There are three important conclusions: a) possibility based classifiers are capable of recognising highly non-linear data, as for example the spiral benchmark; b) possibility generation algorithms should be noise resistant, especially for temporal data; and c) in particular, MPFS architecture offers computational and speed advantages for pattern recognition. Following the investigation of similar issues addressed in previous studies on noise, this paper has documented the results of MPFS on spiral data which may now be used to compare its performance with related classifiers on the same benchmark. If we can establish that possibility based classifier are universal approximators with a generic framework, then several real-time applications can benefit. Hopefully, this study will stimulate further work in the same direction.

REFERENCES

1. Singh, S. and Steinl, M. "Fuzzy Search Techniques in Knowledge-Based Systems", *Proc. 6th International Conference on Data and Knowledge Systems for Manufacturing and Engineering (DKSME'96)*, Tempe, Arizona, pp. 1-10, (1996).
2. S. Singh and M. Steinl, "Fuzzy pattern recognition for knowledge-based systems," *Proceedings of the International conference on knowledge based computer systems*, Bombay, Narosa, pp. 383-394, (1996).
3. S. Singh, E. L. Hines and J. Gardner, "Classifier systems based on possibility distributions: A comparative study," *Proceedings of the 3rd International conference on neural networks and genetic algorithms (ICANNGA97)*, Norwich, Wien: Springer-Verlag, pp.537-540, (1997).
4. *Fuzzy reasoning and its applications*, E. H. Mamdani and B. R. Gaines (eds.), Academic Press, (1981).
5. L. A. Zadeh, *Fuzzy logic and its applications*, Academic press, New York, (1965).
6. S. Singh, E. L. Hines and J. Gardner, "Fuzzy neural computing of coffee and tainted water data on electronic nose," *Sensors and Actuators B*, vol. **30**, issue 3, pp. 190-195, (1996).
7. C. M. Bishop, *Neural networks for pattern recognition*, Clarendon Press, Oxford, (1995).
8. S. M. Weiss, and C. A. Kulikowski, *Computer systems that learn*, Morgan Kaufmann, San Mateo, CA, (1991).
9. M. Brown and C. A. Harris, "Fuzzy logic expert system for iron ore processing," *Proceedings of the IEEE industry applications conference*, Canada, vol. **3**, pp. 2190-9, (1993).
10. S. E. Fahlman, "Faster-learning variations on back-propagation: An empirical study," *Proceedings of the 1988 Connectionist models summer school*, Morgan Kaufmann, (1988).
11. S. E. Fahlman and C. Lebiere, "The Cascade-Correlation learning architecture," In Touretzky (Ed.) *Advances in neural information processing systems 2*, Morgan Kaufmann, (1990).
12. K. J. Lang and M. J. Witbrock, "Learning to tell two spirals apart," *Proceedings of the 1988 Connectionist models summer school*, Morgan Kaufmann, (1988).
13. D. S. Touretzky and D. A. Pomerleau, "What's hidden in the hidden layers ?" *Byte* (August issue), pp. 227-233, (1989).

14. T. Chin-Chi and B. W. Wah, "An automated design system for finding the minimal configuration of a feed-forward neural network," *Proceedings of the IEEE International conference on neural networks*, Florida, vol. **3**, pp. 1295-1300, (1994).

15. H. C. Chua, J. Jia, L. Chen and Y. Gong, "Solving the two-spiral problem through input data encoding," *Electronics letters*, vol. **31**, issue 10, pp. 813-14, (1995).

16. J. Jia and H. Chua, "Solving two-spiral problem through input data representation," *Proceedings of the IEEE International conference on neural networks*, vol. **1**, pp. 132-135, (1995).

17. C. T. Sun and J. S. Jang, "A neuro-fuzzy classifier and its applications," *Proceedings of the IEEE International conference on fuzzy systems*, vol. **1**, pp. 94-98, (1993).

18. L. P. Tay and D. J. Evans, "Fast learning artificial neural network (FLANN II) using the nearest neighbour recall," *Neural, parallel and scientific computations*, vol. **2**, issue 1, pp.17-27, (1994).

19. F. Ulgen, N. Akamatsu and T. Iwasa, "The hypercube separation algorithm: a fast and efficient algorithm for on-line handwritten character recognition," *Applied Intelligence*, vol. **6**, issue 2, pp. 101-116, (1996).

20. R. M. Burton, and G. J. Mpitsos, "Event-dependent control of noise enhances learning in neural networks," Neural Networks, vol. **5**, pp. 627-637, (1992).

21. K. Jim, B. Horne and C. L. Giles, "Effects of noise on convergence and generalisation in recurrent networks," *Neural Information Processing Systems* 7, G. Tesauro, D. Touretzky and T. Leen (eds.), MIT Press, p. 649, (1995).

22. A. F. Murray and P. J. Edwards, "Synaptic weight noise during multilayer perceptron training: Fault tolerance and training improvements," *IEEE Transactions on Neural Networks*, vol. **4**, issue 4, pp. 722-725, (1993).

23. G. C. Canavos, "*Applied probability and statistical methods*," Little Brown and Company, (1984).

24. B. Kosko, *Neural networks and fuzzy systems - A dynamical systems approach to Machine Intelligence*, Prentice Hall International edition, (1992).

Figure 1 The Massively Parallel Fuzzy System (MPFS).

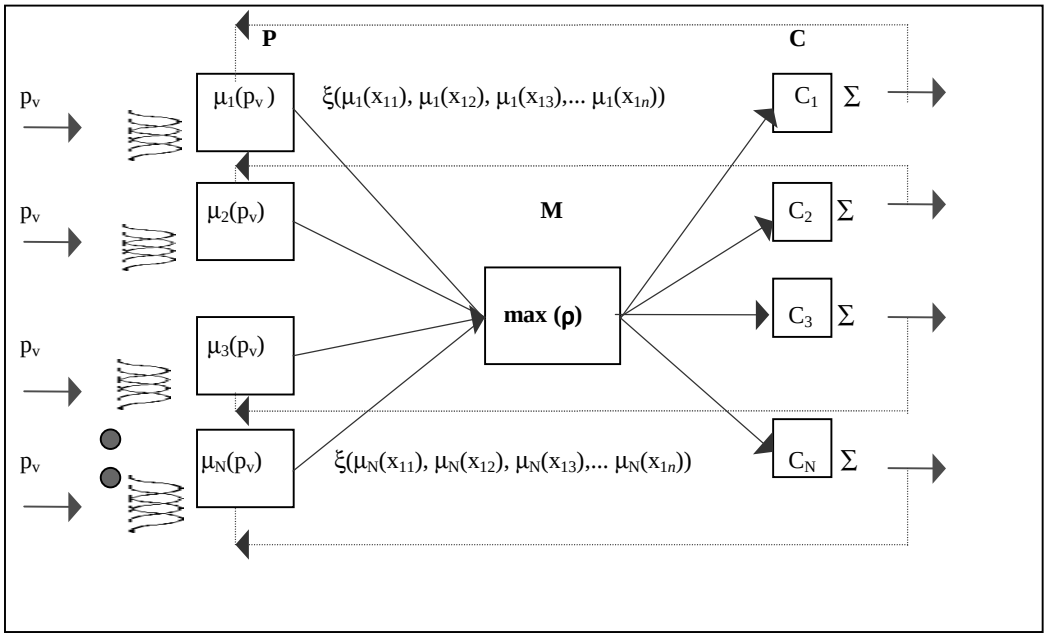

Figure 2. 2D Spiral data scatterplot. Two spirals with a maximum radius of 6.5 coil around each other. The two different classes are highlighted in a hypersphere with their training data (white and black points) and a test pattern is illustrated with a black square.

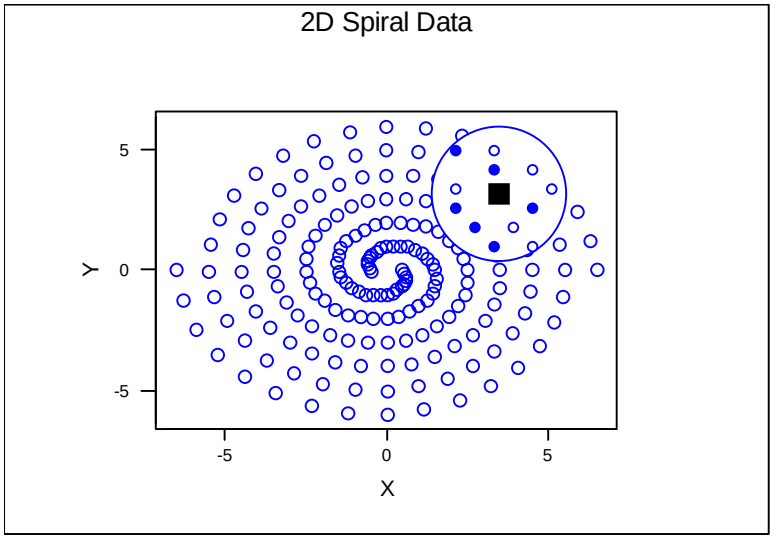

Figure 3. The effect of standard deviation on the generalisation error % with a maximum offset δ = 4.1(*Gaussian* noise). NC(m) ≡ non-cumulative *multiplicative*; NC(a) ≡non-cumulative *additive*.

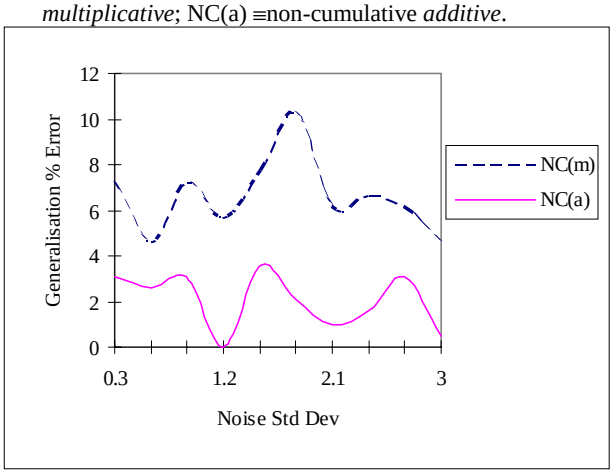

Figure 4. The effect of standard deviation on the generalisation error % with a maximum offset δ = 4.1 (*lognormal noise*). NC(m) ≡ non-cumulative *multiplicative*; NC(a) ≡ non-cumulative *additive*.

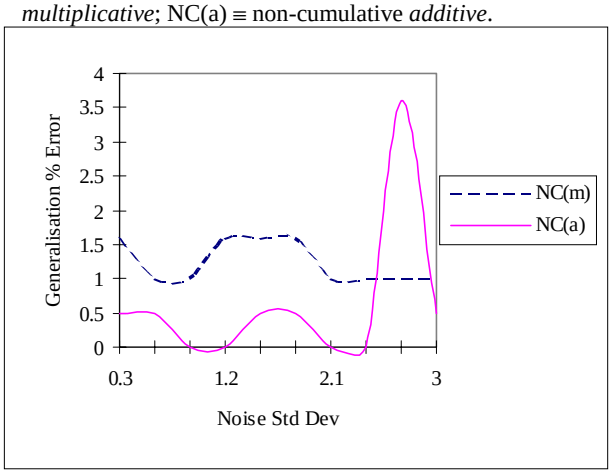

Figure 5. The effect of standard deviation on the generalisation error % with a maximum offset δ = 4.1(*Gaussian* noise). C(m) ≡ cumulative *multiplicative*; C(a) ≡ cumulative *additive*.

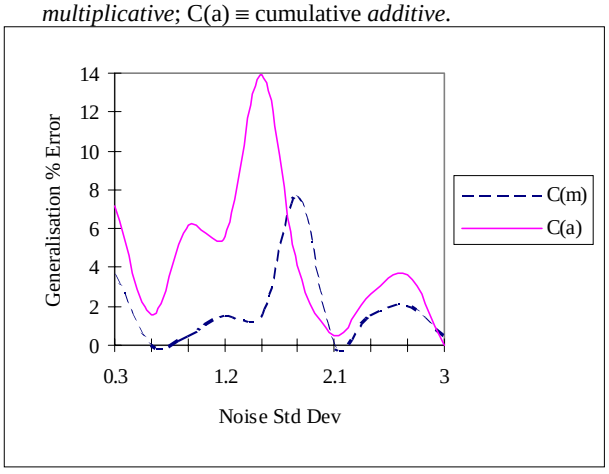

Figure 6. The effect of standard deviation on the generalisation error % with a maximum offset δ = 4.1 (*lognormal noise*). C(m) ≡ cumulative *multiplicative*; C(a) ≡ cumulative *additive*.

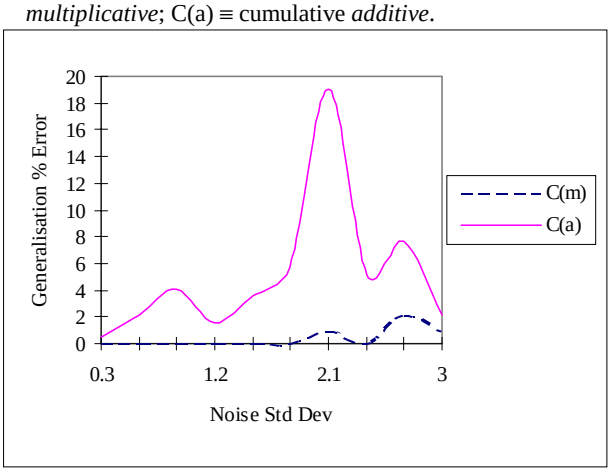

Table 1. Noise distribution summary for vector **r** generation
The distributions are described by their variance σ^2 and mean μ

Noise Distribution	*Description*	*Characteristics*
Uniform	p(r) = 1.0 for all random variables	range [0, 1]
Gaussian	p(r) = $\frac{1}{(2\pi\sigma^2)^{1/2}} \exp\{\frac{(r-\mu)^2}{2\sigma^2}\}$	σ^2 = 1.0, μ = 0.0, range [-1, 1] Here μ is the mean and σ^2 is the variance
LogNormal	log(r) is normally distributed as above	σ^2 = 1.0, μ = 0.15, range [0, 1]
Negative Exponential	p(r) = $\frac{1}{\mu}\exp(-r/\mu)$ for r > 0 and μ > 0 else p(r) = 0	range [0, 1] μ = .15
Erlang	p(r) = $\frac{(\mu k)^k}{(k-1)!}$ .r^{k-1} $e^{-k\mu r}$	$k = 1/\mu.\sigma^2$, μ = 0.5, σ^2 = 1.0, range [0, 1]

Table 2. The performance of the MPFS system with varying *additive* noise: Recognition rate ℜ % with increasing offset limit δ

Offset δ	*Uniform*		*Gaussian*		*LogNormal*		*Neg Exp*		*Erlang*	
	T	V	T	V	T	V	T	V	T	V
.1	99.0	100	99.0	100	99.0	100	99.0	100	99.0	100
.5	99.5	100	99.0	100	99.0	100	99.0	100	99.0	100
.9	100	100	97.9	100	99.0	100	99.0	100	98.4	99.0
1.3	99.5	99.5	97.4	99.0	99.0	100	99.0	100	99.0	99.0
1.7	99.5	99.5	97.9	96.4	98.4	99.0	98.45	100	97.4	98.4
2.1	97.9	99.5	96.9	94.3	98.4	100	99.0	100	98.4	98.4
2.5	97.4	99.5	95.4	94.8	99.0	99.5	99.0	100	95.4	97.9
2.9	95.9	99.5	90.7	93.3	99.0	100	99.0	100	95.4	96.9
3.3	93.3	94.8	91.2	93.3	97.4	99.5	99.0	100	95.9	97.4
3.7	93.8	94.8	89.7	88.1	97.4	98.4	99.5	99.5	96.4	95.9
4.1	93.3	94.3	89.7	87.1	97.9	97.9	97.9	100	93.8	97.4

Table 3. The performance of the MPFS system with varying *multiplicative* noise: Recognition rate ℜ % with increasing offset limit δ

Offset δ	*Uniform*		*Gaussian*		*LogNormal*		*Neg Exp*		*Erlang*	
	T	V	T	V	T	V	T	V	T	V
.1	98.9	100	99.0	100	99.0	100	99.0	100	99.0	100
.5	98.9	99.5	97.9	97.9	99.0	100	99.0	100	99.0	99.5
.9	98.9	99.0	96.4	93.3	99.0	100	99.0	99.5	99.0	99.5
1.3	97.9	97.9	92.8	88.1	98.4	100	99.0	99.0	99.0	99.0
1.7	98.4	99.5	91.7	92.8	98.4	100	99.0	100	97.9	97.9
2.1	96.9	96.9	89.2	88.6	98.4	100	99.0	99.5	98.9	96.9
2.5	97.9	97.4	92.8	91.7	99.0	100	99.0	99.0	96.9	99.5
2.9	97.4	97.9	90.2	88.6	98.4	100	99.0	100	97.4	99.0
3.3	94.8	98.4	94.3	93.8	98.4	97.9	99.5	98.4	99.0	96.4
3.7	95.3	94.9	89.2	88.5	97.4	99.0	98.4	97.9	97.9	96.9
4.1	95.9	95.9	93.3	86.1	98.9	99.5	97.42	99.5	96.9	96.4

Table 4. The performance of the MPFS system with varying noise: Entropy change with *uniform* additive (*a*) and multiplicative (*m*) noise

Offset δ	*Entropy (m)*		*Av. Total Noise*$\tilde{N}$ *(m)*		*Entropy (a)*		*Av. Total Noise*$\tilde{N}$ *(a)*
	T	V	T	V	T	V	T or V
.1	284	285	.1	.1	289	288	.05
.5	265	265	.57	.57	276	276	.25
.9	253	249	.96	1.00	265	265	.44
1.3	249	245	1.35	1.45	259	261	.62
1.7	240	233	2.08	1.93	251	254	.87
2.1	238	230	2.38	2.38	249	248	1.00
2.5	233	225	2.78	2.89	244	242	1.24
2.9	232	220	3.24	3.30	241	240	1.42
3.3	230	213	3.86	3.86	240	237	1.67
3.7	225	210	4.34	4.36	233	231	1.95
4.1	230	210	4.57	4.48	237	236	2.05

Table 5. The performance of the MPFS system with varying noise:
Entropy change with *Gaussian* additive (*a*) and multiplicative (*m*) noise

Offset δ	*Entropy (m)*		*Av. Total NoiseÑ (m)*		*Entropy (a)*		*Av. Total NoiseÑ (a)*
	T	V	T	V	T	V	T or V
.1	281	282	.18	.19	287	286	.08
.5	257	257	.91	.91	267	270	.41
.9	249	247	1.57	1.58	258	259	.76
1.3	244	240	2.33	2.36	250	253	1.06
1.7	239	227	3.06	3.13	245	244	1.40
2.1	232	221	3.92	3.92	239	241	1.79
2.5	233	216	4.65	4.61	239	236	1.98
2.9	227	217	5.39	5.05	232	225	2.34
3.3	224	204	5.78	6.07	231	223	2.63
3.7	223	204	6.55	6.88	228	221	3.17
4.1	221	203	6.58	7.03	224	217	3.29

Summary

This paper studies the performance of Massively Parallel Fuzzy Systems on noisy spiral data. The paper first details the working of such a system illustrating its architecture. This description is followed by a fuzzy approach to spiral recognition which details the use of possibility computation for assigning the predicted class of test patterns. This algorithm is based on the use of upper and lower bounds of test data in the training set and the distance between test data and its nearest neighbour in the training set. Possibilities for individual datums are combined with a multiplication operator to assign the possibility of a test pattern belonging to a particular class. The above approach is tested on noisy spiral data. Noise injection is first discussed to illustrate the quality and the quantity of noise that contaminates the training data: a total five statistical continuous distributions are used for generating pseudo-random noise. The performance analysis of the classifier is discussed under two separate headings: its behaviour in terms of recognition rate, levels of noise, and system entropy; and its generalisation ability which investigates the change in generalisation error with varying the standard deviation of the noise distribution. These analyses show that the classifier is relatively stable to noise, has a low error rate with varying degrees of random noise, and performs well with high recognition rates. The paper also discusses how system behaviour is different for additive and multiplicative noise, and for cumulative and non-cumulative noise.